

# Classical Unsupervised Machine Learning

John Armstrong



# Different Types of Machine Learning

	Supervised	Unsupervised
Classical	Decision Trees, SVMs, shallow neural networks	Clustering, Dimensionality Reduction
Deep	Multi-layer ANNs/ CNNs, ResNets	GANs, VAEs, INNs

# Different Types of Machine Learning

	Supervised	Unsupervised
Classical	Decision Trees, SVMs, shallow neural networks	Clustering, Dimensionality Reduction
Deep	Multi-layer ANNs/ CNNs, ResNets	GANs, VAEs, INNs

- Unsupervised = allowing the computer to identify the important features in your data.
- This is beneficial for identifying patterns or important quantities in your data.
- Prior knowledge of the dataset isn't required!

# How does unsupervised learning work?

$$f(\overrightarrow{X}, \{a_1, \dots, a_n\}) = \overrightarrow{Y}$$

# How does unsupervised learning work?

$$f(\vec{X}, \{a_1, \dots, a_n\}) = \vec{Y}$$

## Input Data:

- Raw data
- Extracted features
- Distances/  
correlations



# How does unsupervised learning work?

$$f(\vec{X}, \{a_1, \dots, a_n\}) = \vec{Y}$$

## Input Data:

- Raw data
- Extracted features
- Distances/  
correlations

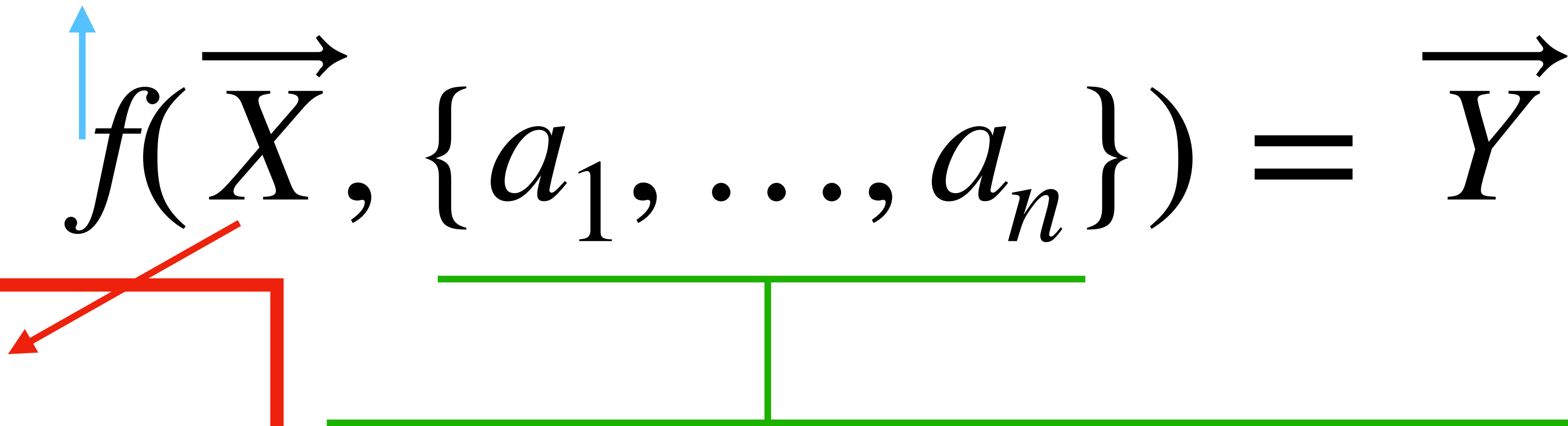
## Hyperparameters:

- Tuning parameters for the algorithm.
- The supervised bit of unsupervised learning

# How does unsupervised learning work?

## Internal Choices of the computer:

- How the computer manipulates the input and the hyperparameters

$$f(\vec{X}, \{a_1, \dots, a_n\}) = \vec{Y}$$


## Input Data:

- Raw data
- Extracted features
- Distances/  
correlations

## Hyperparameters:

- Tuning parameters for the algorithm.
- The supervised bit of unsupervised learning



# How does unsupervised learning work?

## Internal Choices of the computer:

- How the computer manipulates the input and the hyperparameters

$$f(\overrightarrow{X}, \{a_1, \dots, a_n\}) = \overrightarrow{Y}$$


## Output:

- The output of the algorithm

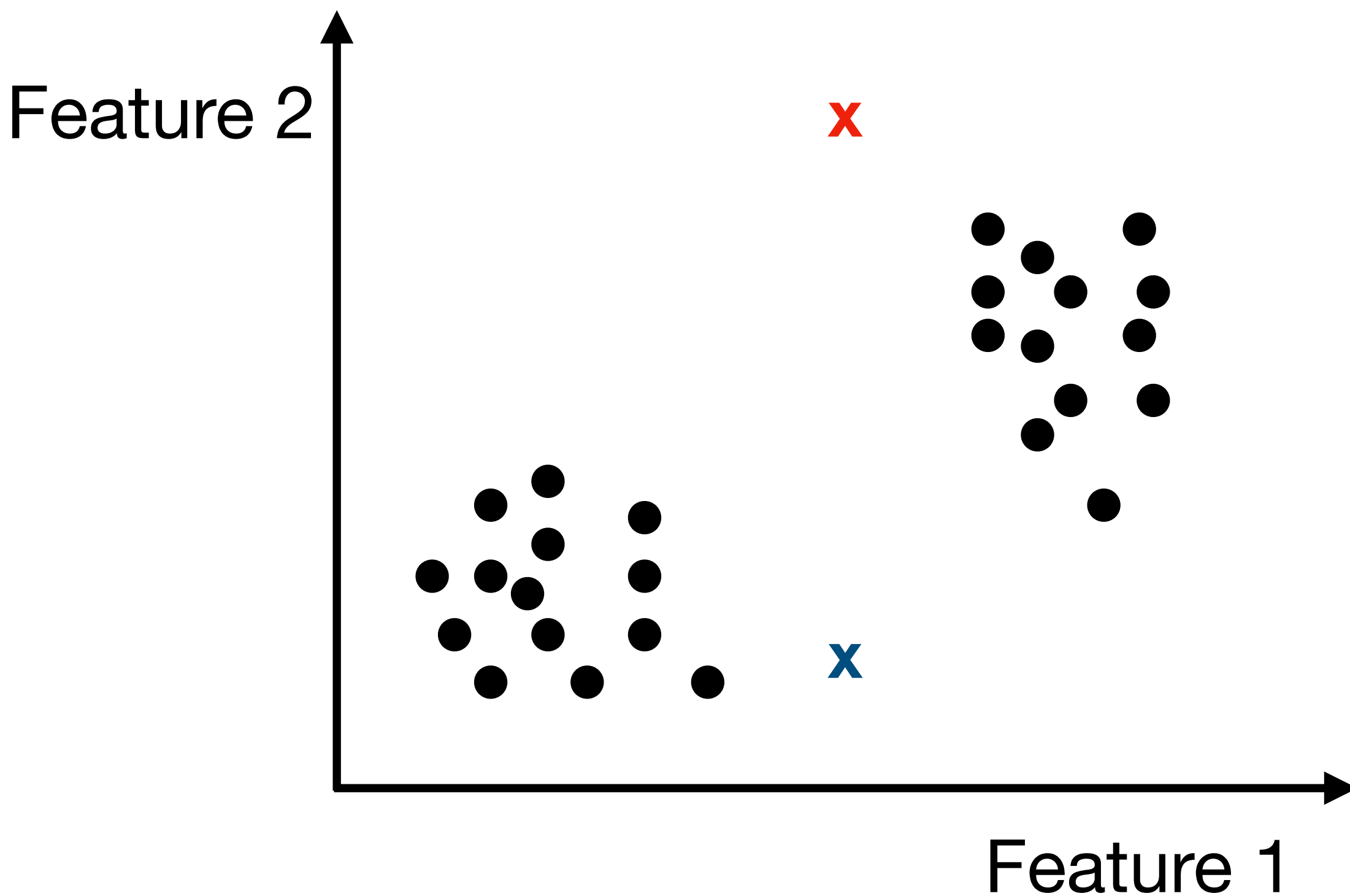
## Input Data:

- Raw data
- Extracted features
- Distances/  
correlations

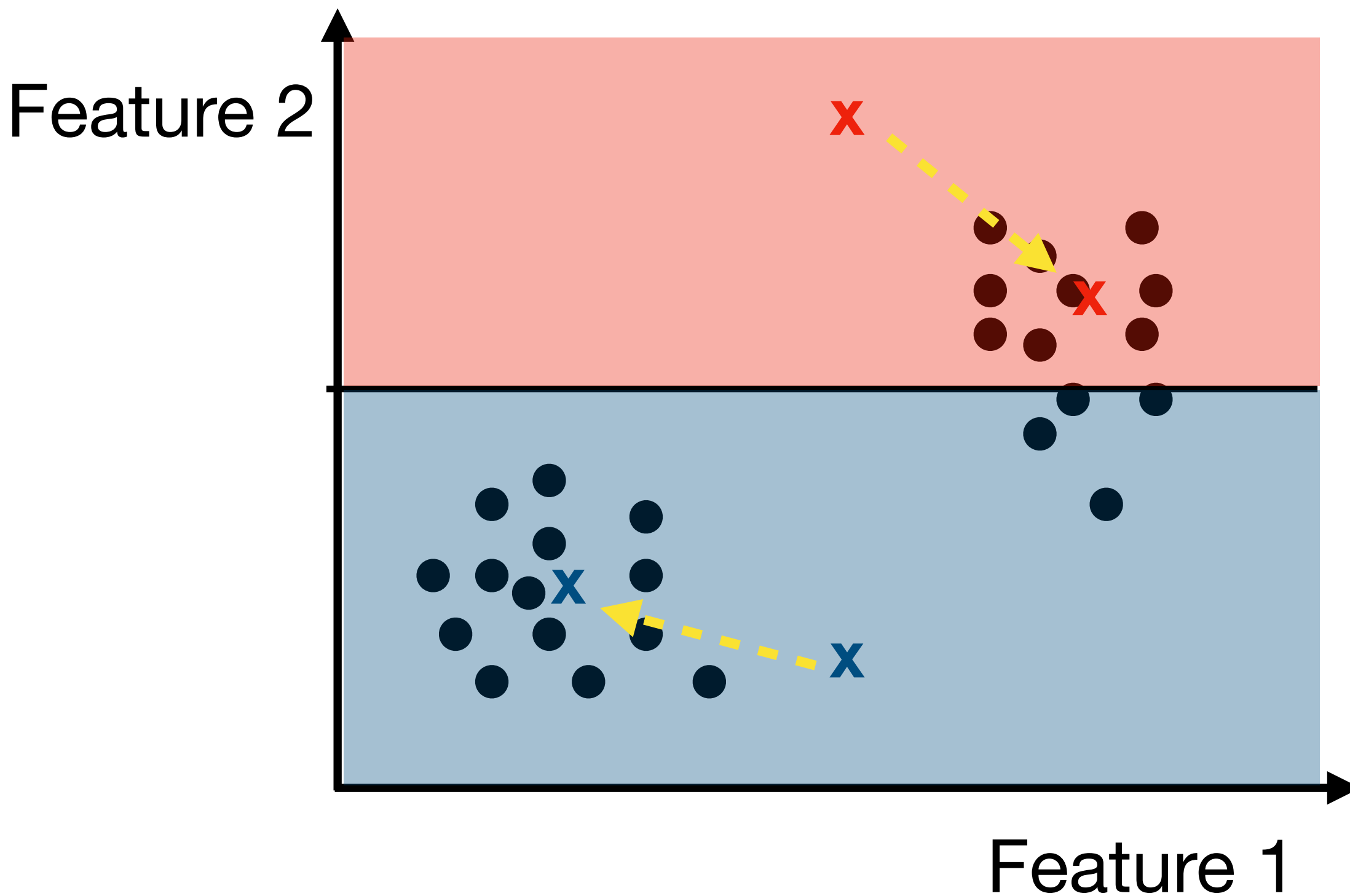
## Hyperparameters:

- Tuning parameters for the algorithm.
- The supervised bit of unsupervised learning

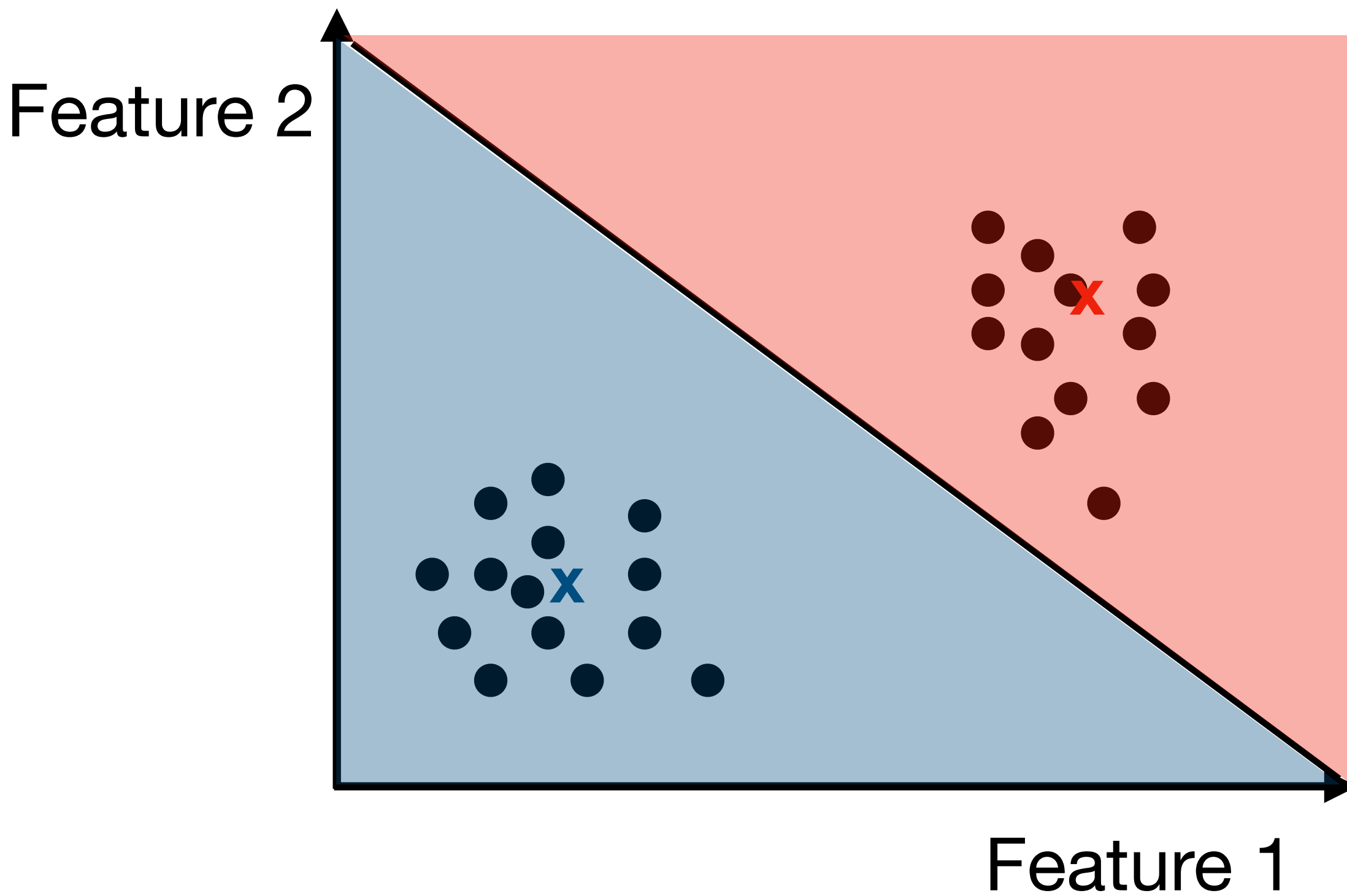
- Groups together data points with similar characteristics
- The methods for doing this typically involve distances in the plane of your observables



- One hyperparameter to be tuned: the number of clusters,  $k$ .
- Start with  $k$  centroids of clusters which are randomly initialised in the plane of the features
- Assign each point to a cluster based on the minimum distance to the centroid



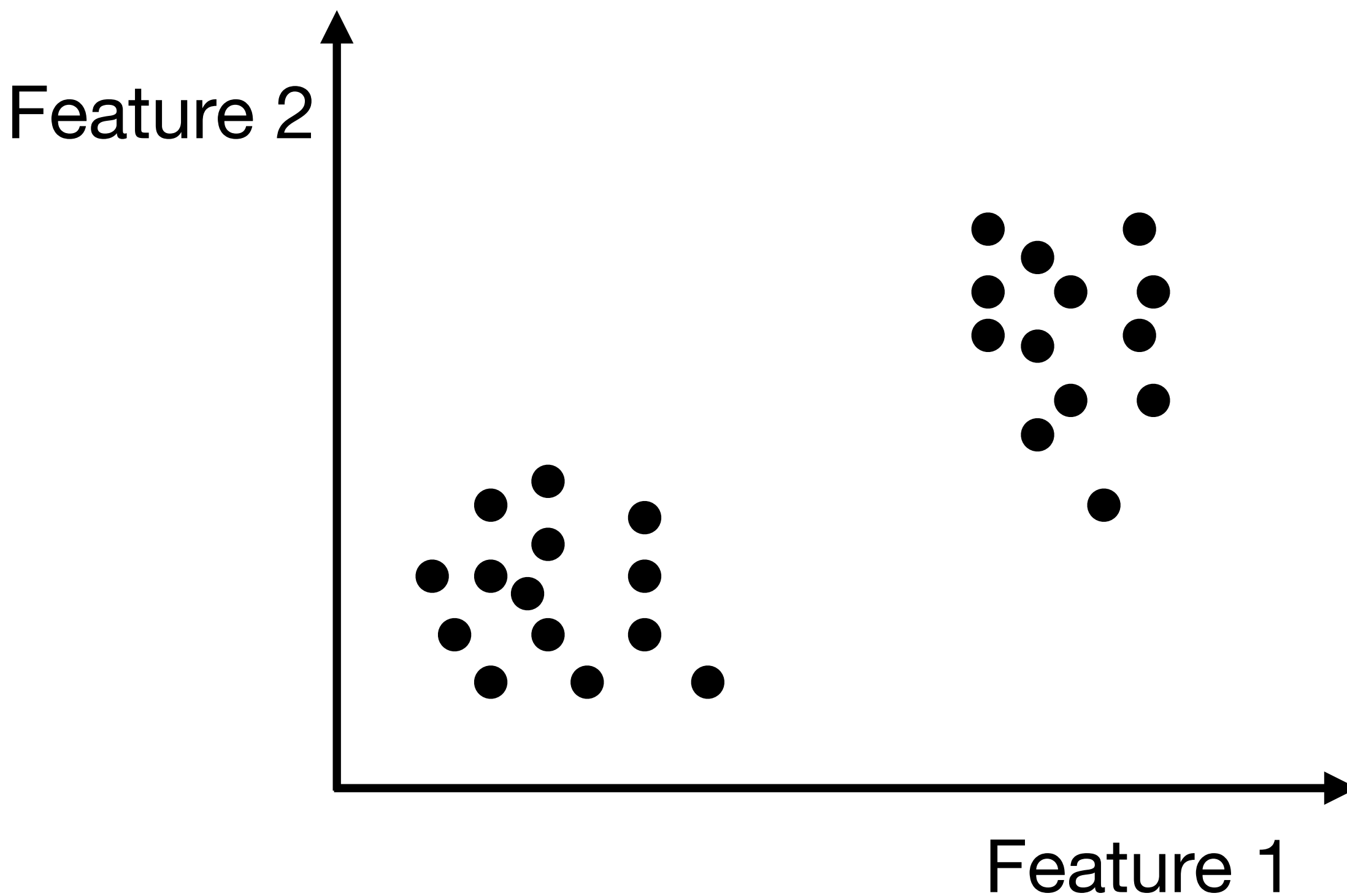
- Calculate the average of the points in the cluster region and make this average the new centroid
- Adjust the decision boundary accordingly by calculating the distances to the new centroids of every point



- We judge the algorithm to be converged to the number of clusters after the centroid changing after each iteration falls below a certain threshold

## The method is simple:

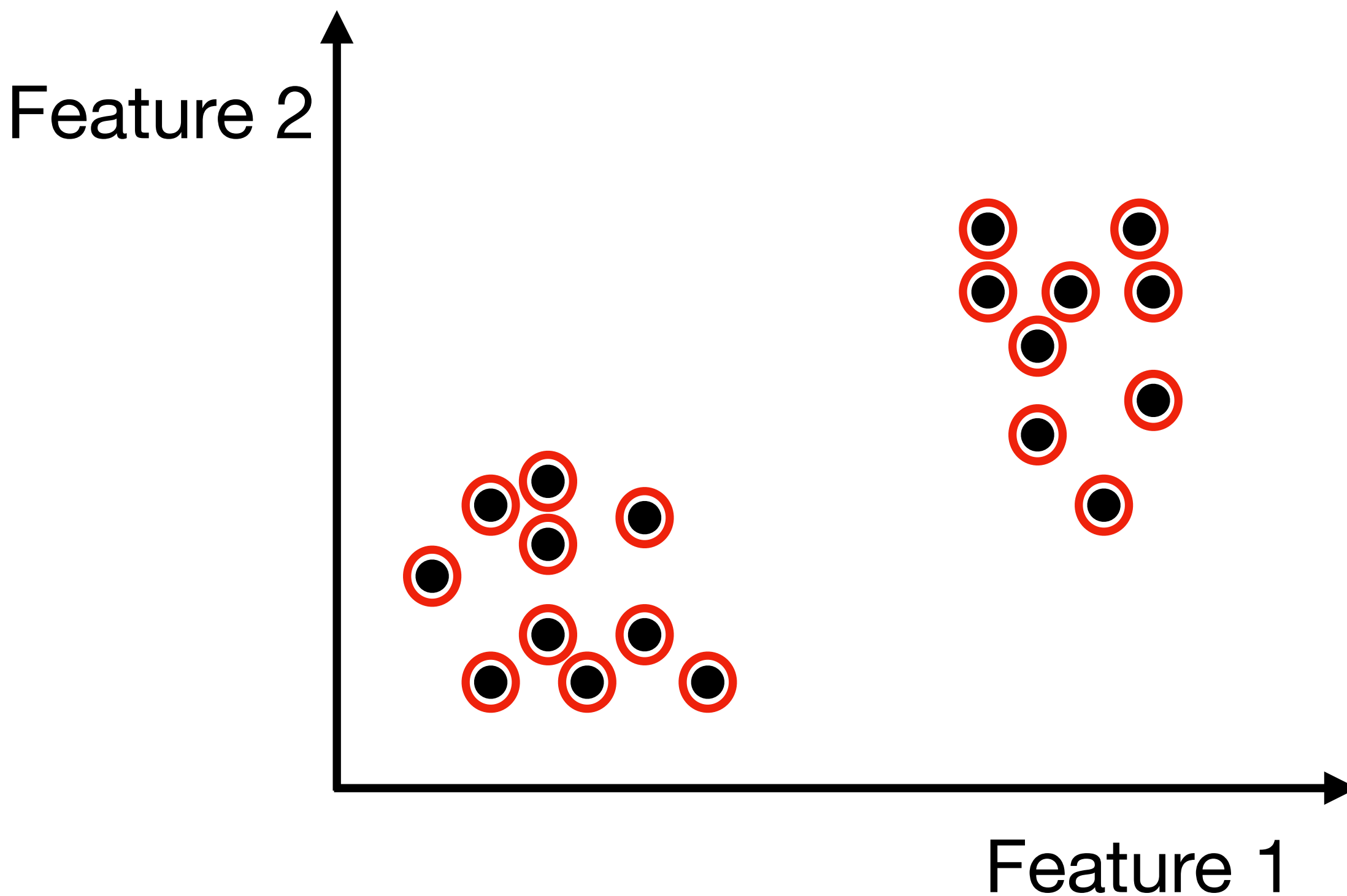
- Each point starts as its own cluster
- The distances between the clusters are calculated
- The two closest clusters are merged into one cluster
- The process is repeated until all the points are in one large cluster



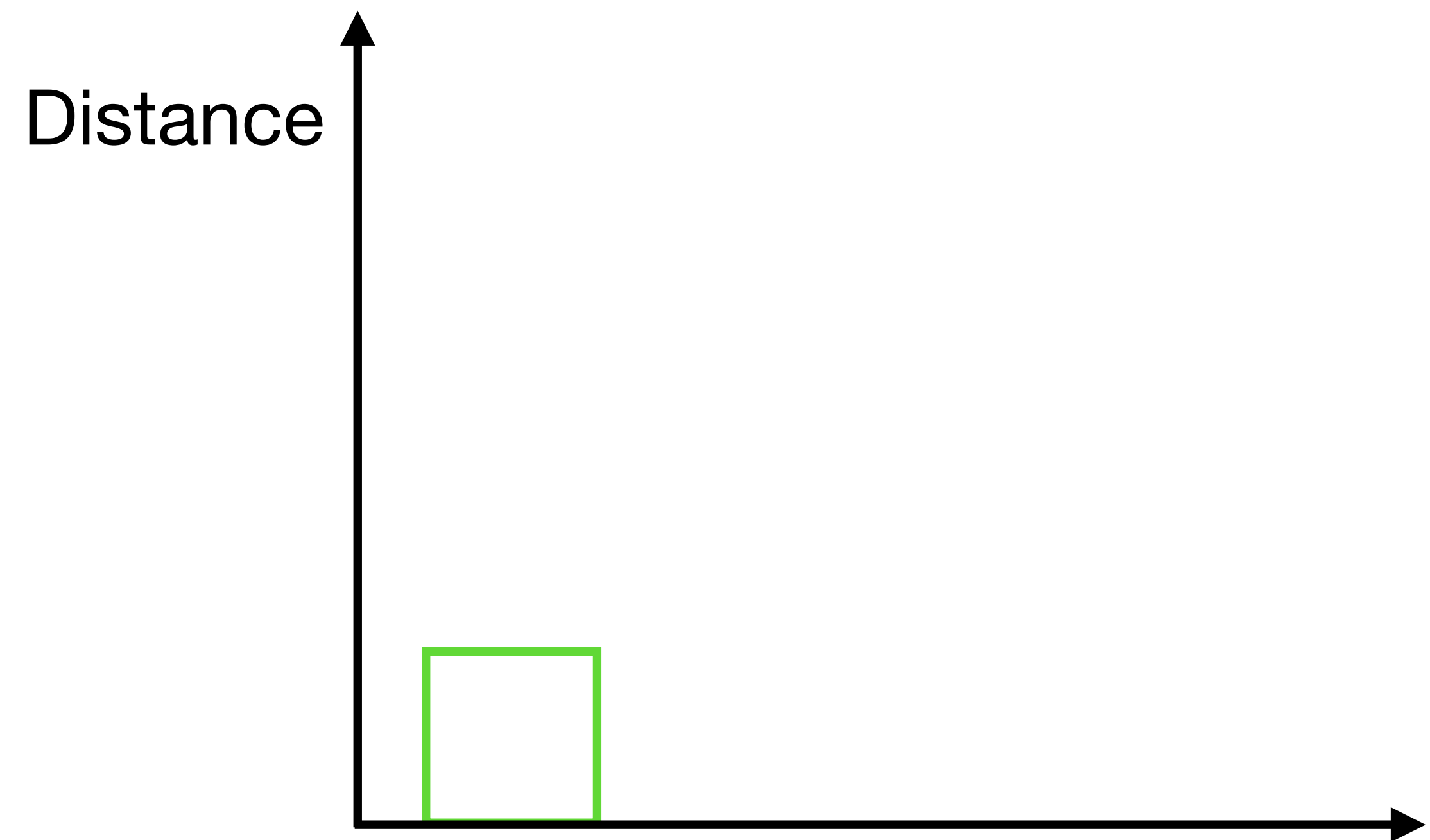
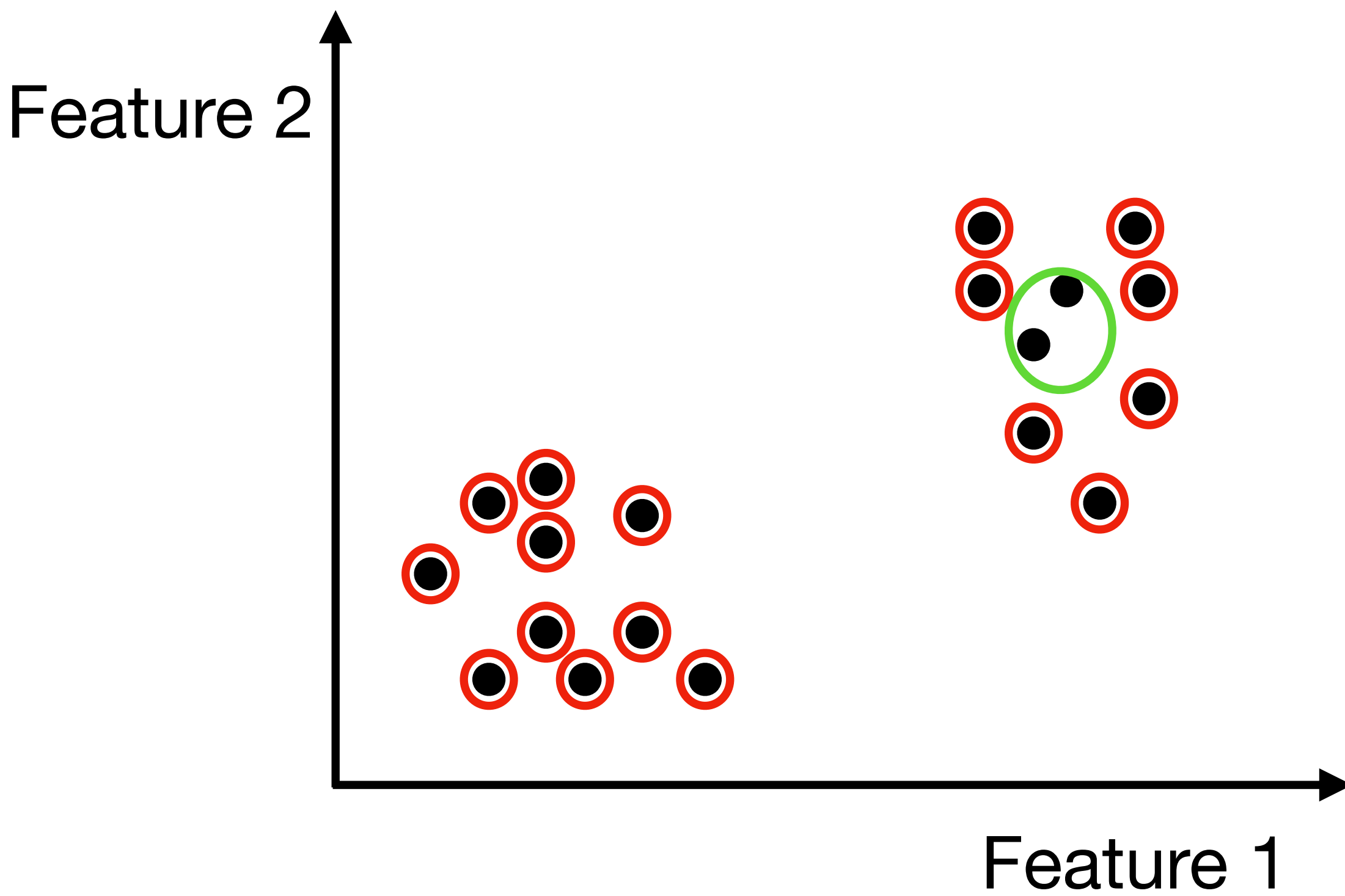


## The method is simple:

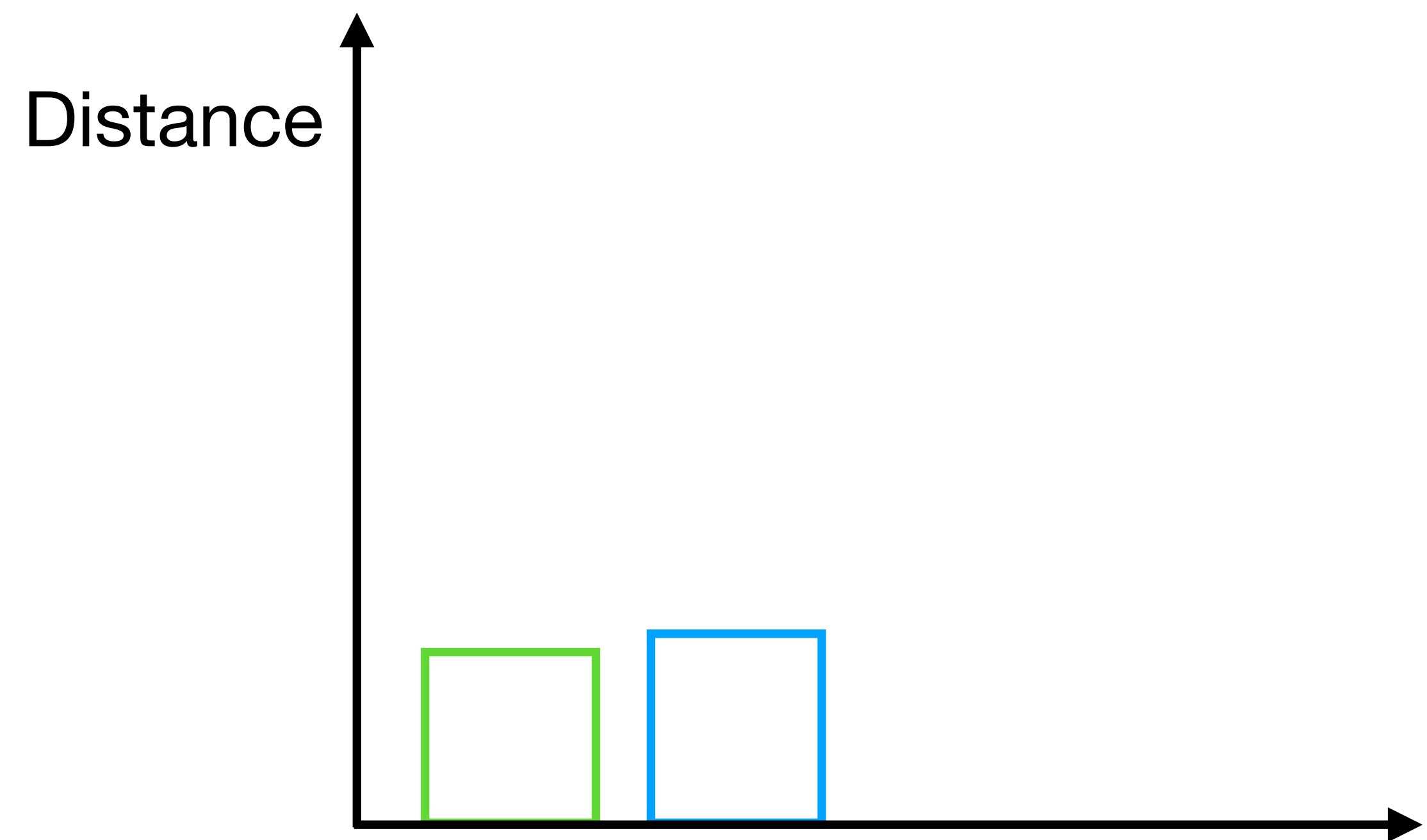
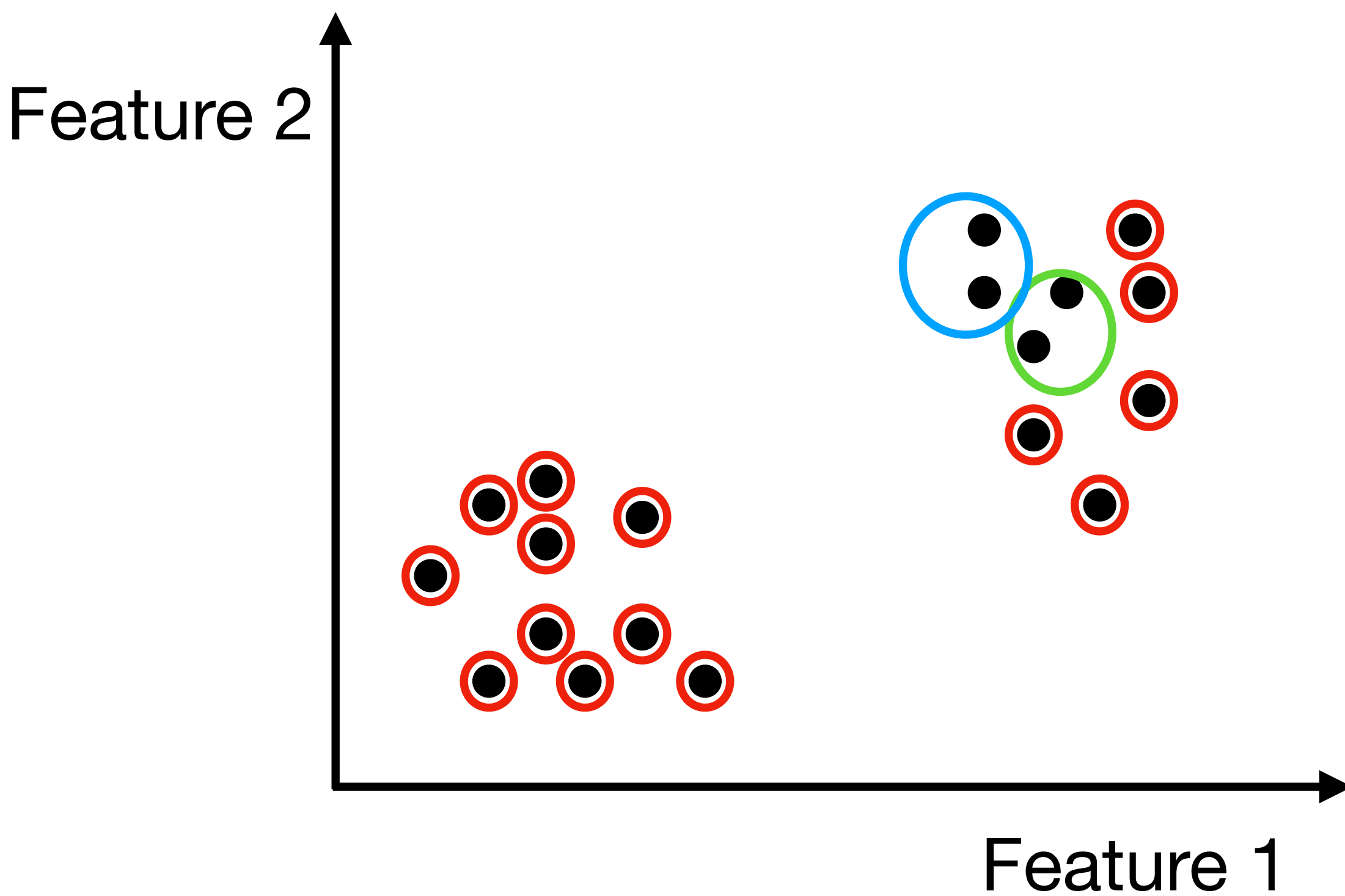
- Each point starts as its own cluster
- The distances between the clusters are calculated
- The two closest clusters are merged into one cluster
- The process is repeated until all the points are in one large cluster



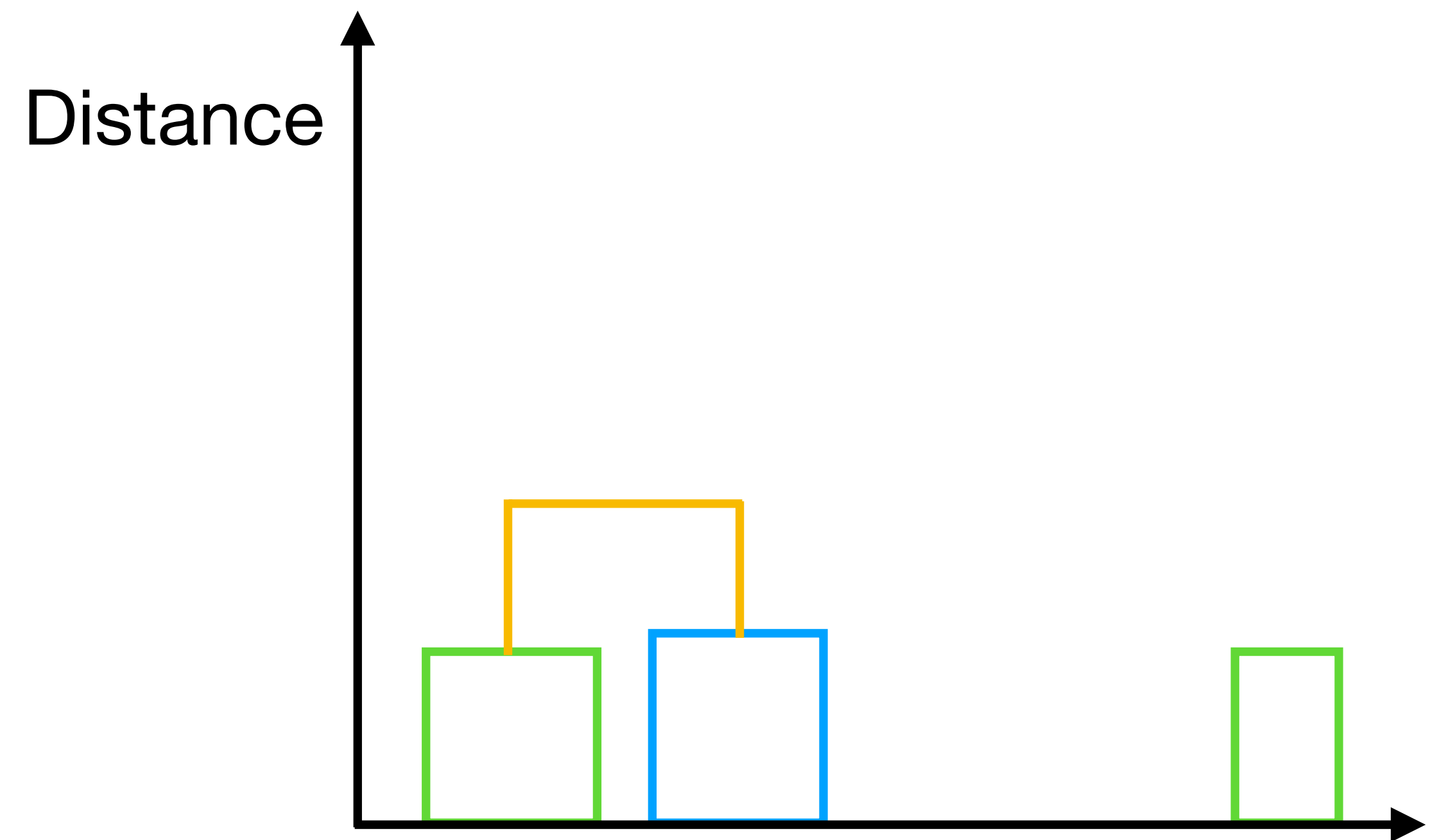
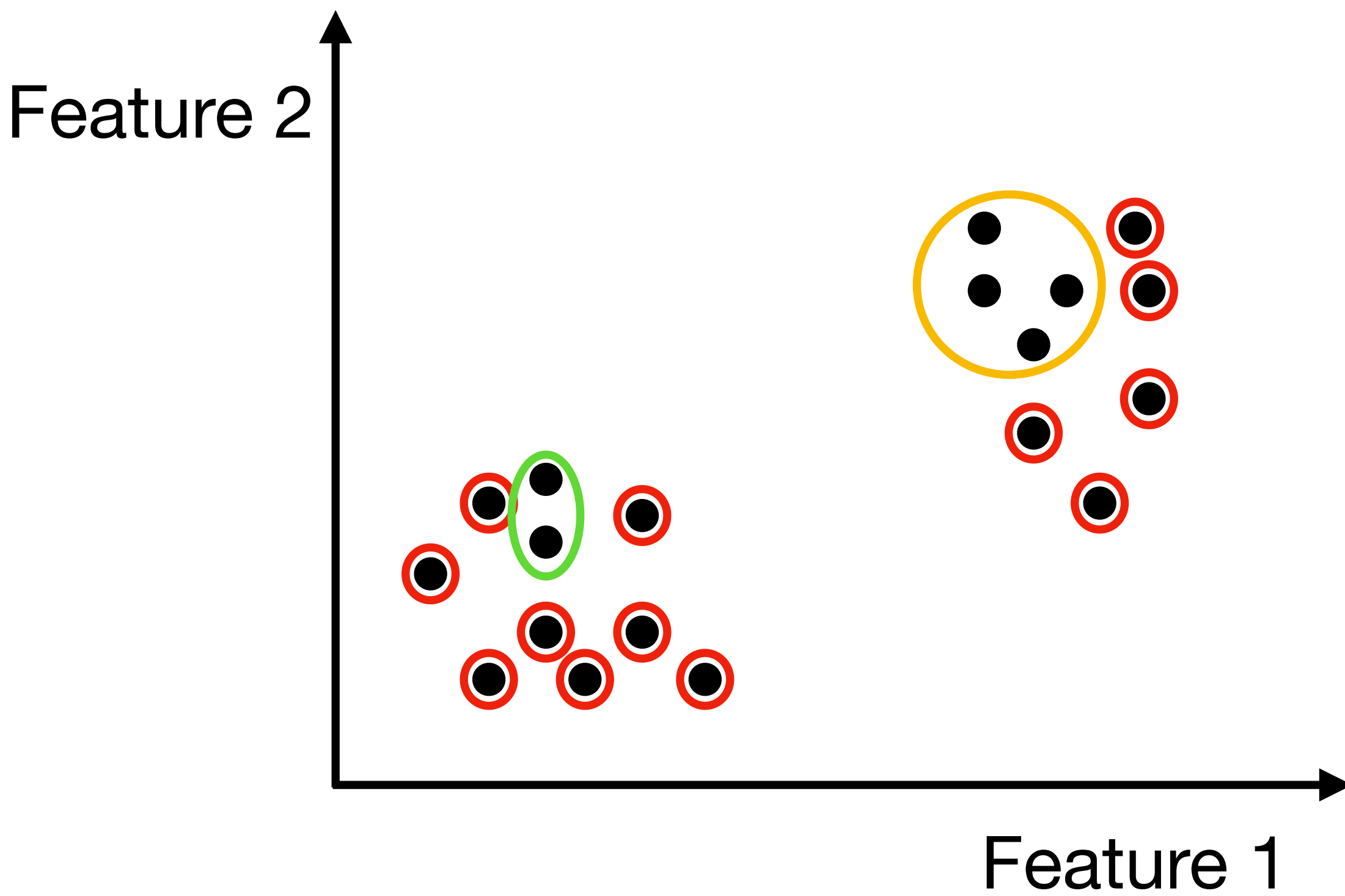
## Dendrogram



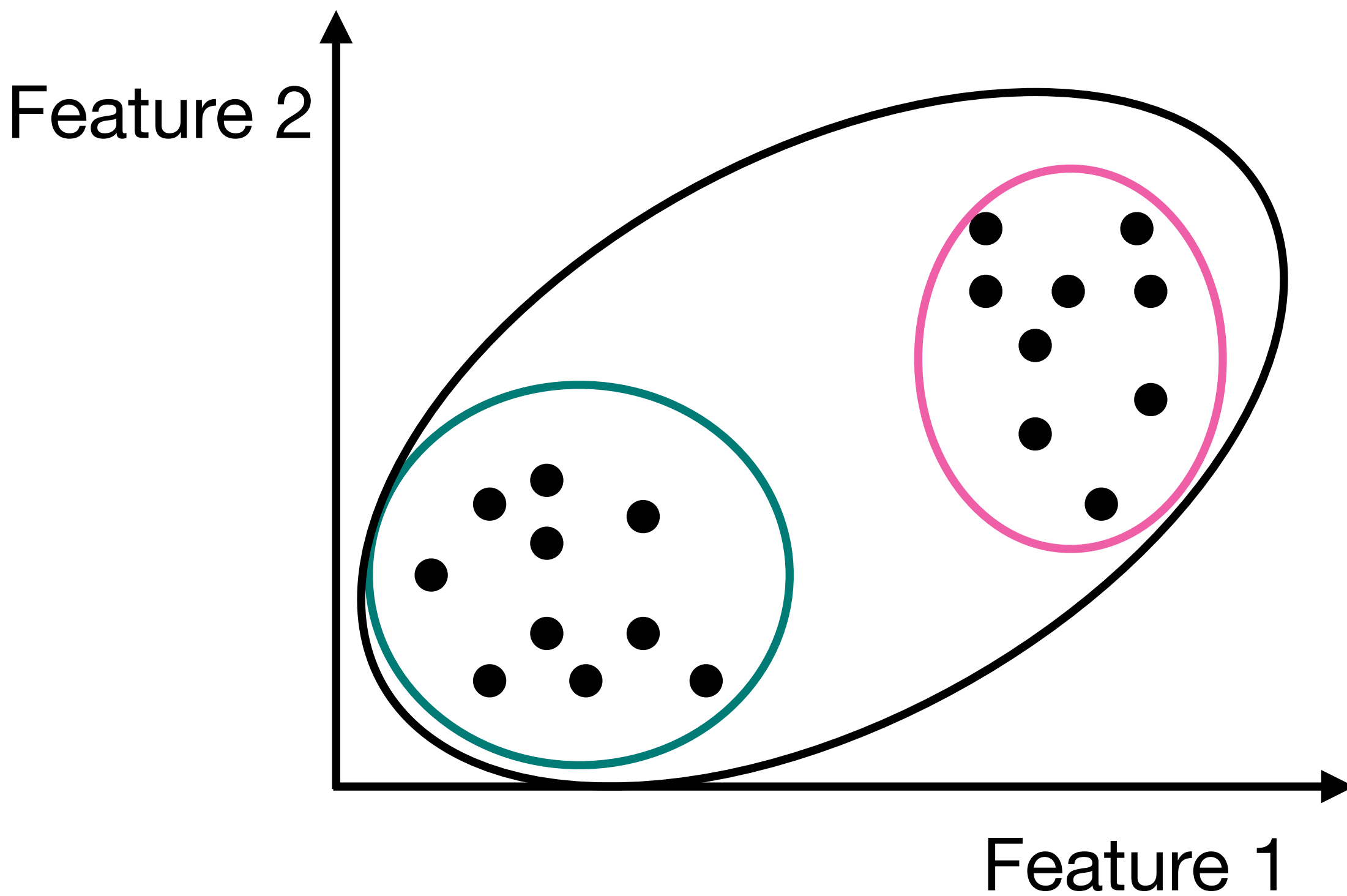
## Dendrogram



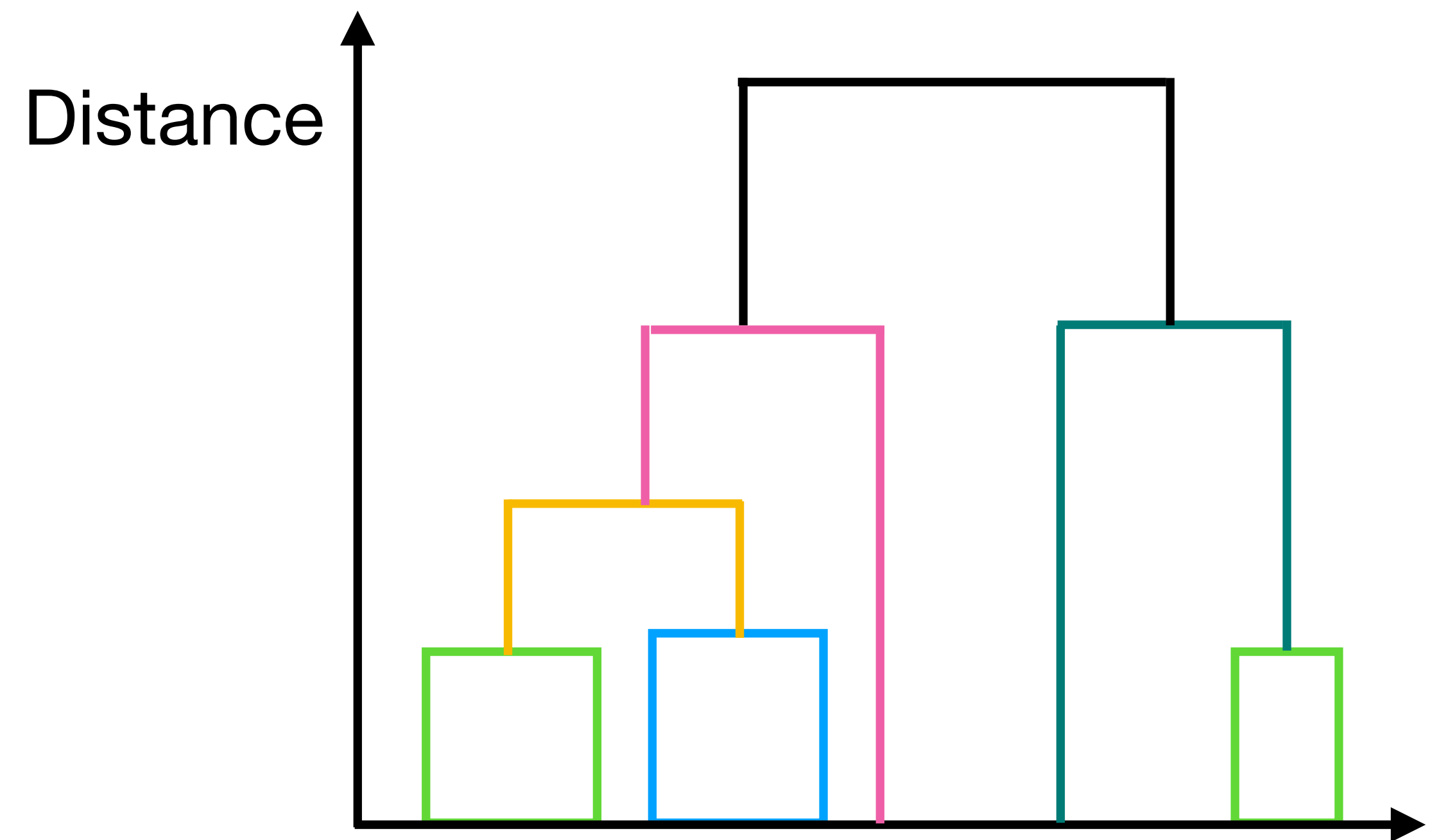
## Dendrogram

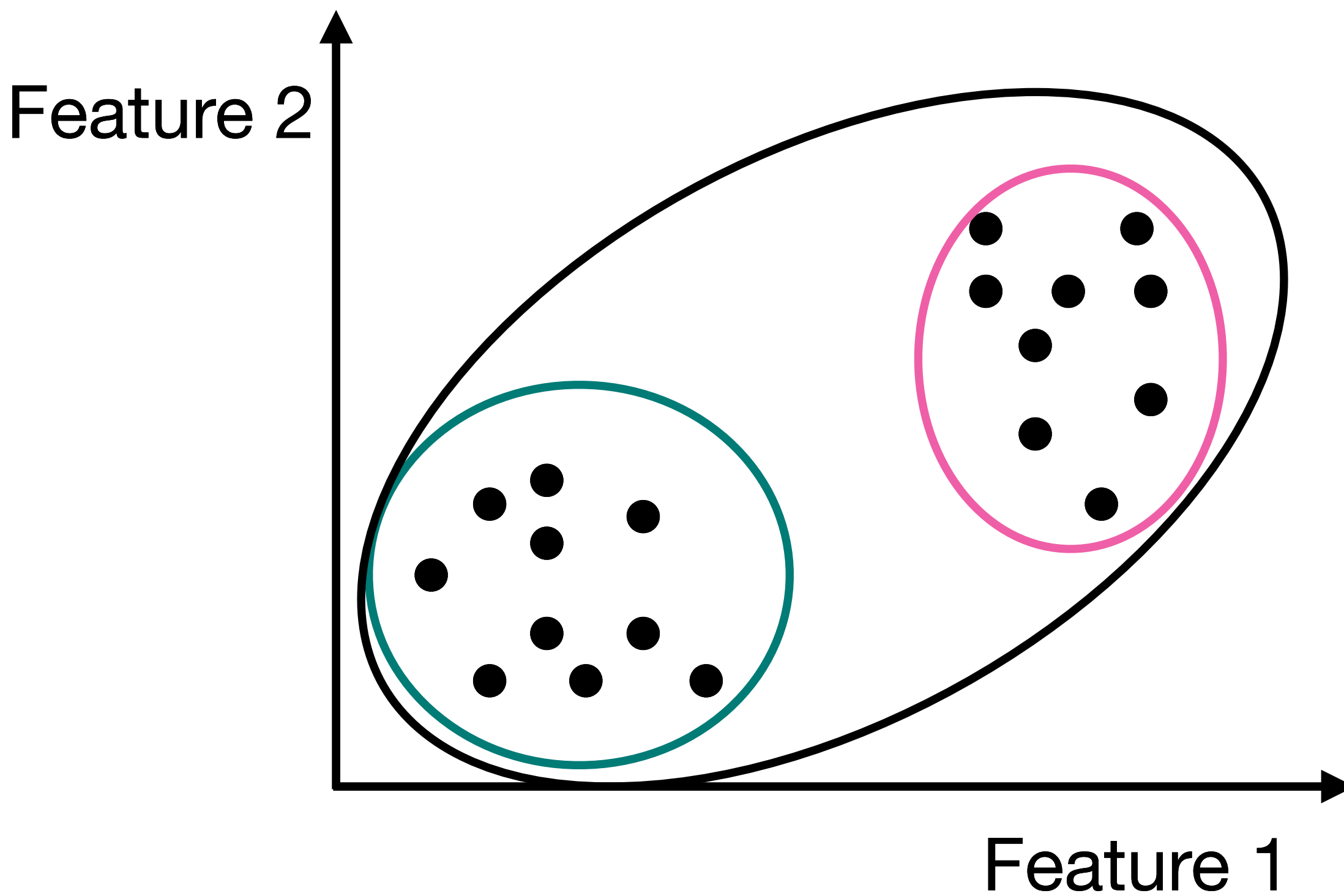


etc....



## Dendrogram

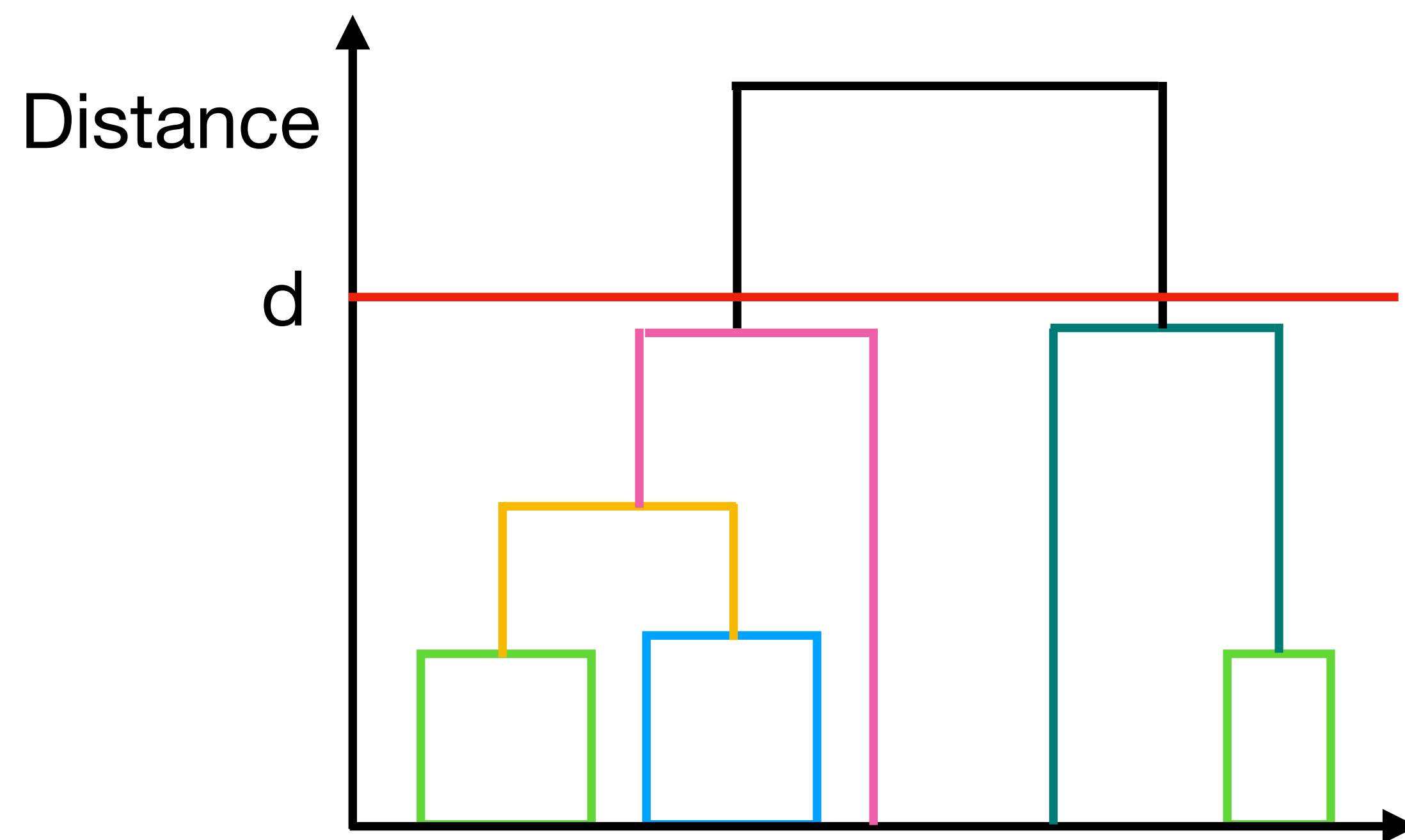




## There are 3 hyperparameters:

1. The number of clusters,  $k$   
e.g. in our example, it is obvious that  $k = 2$ ,  
however in reality, it may be more difficult and  
you may need to experiment with  $k$   
Tuning  $k$  can be done by changing the cutoff  
distance  $d$  in your dendrogram

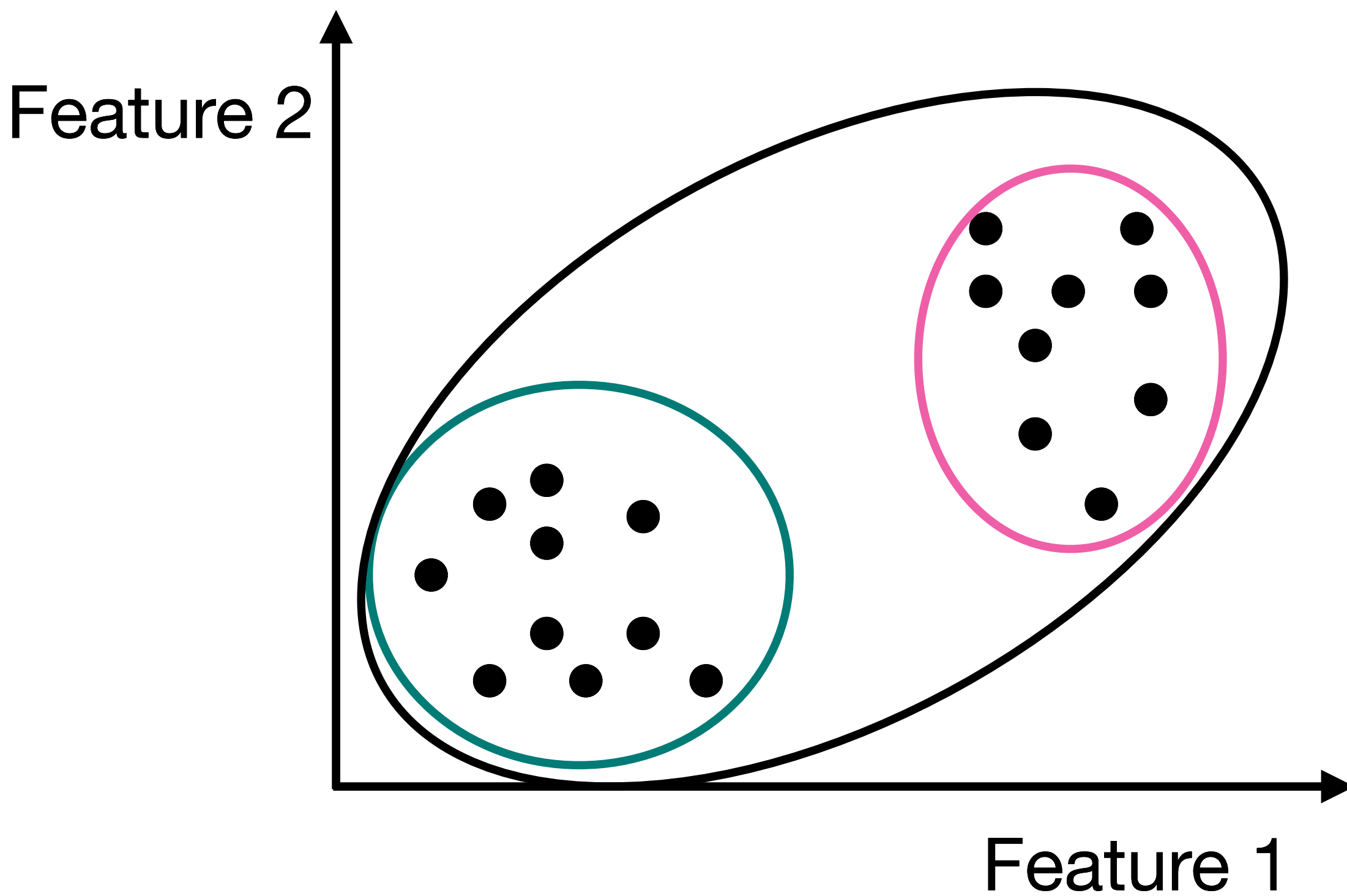




**There are 3 hyperparameters:**

2. Cutoff distance,  $d$

This is the distance at which you decide there is too large a distance between clusters for them to be part of one larger cluster



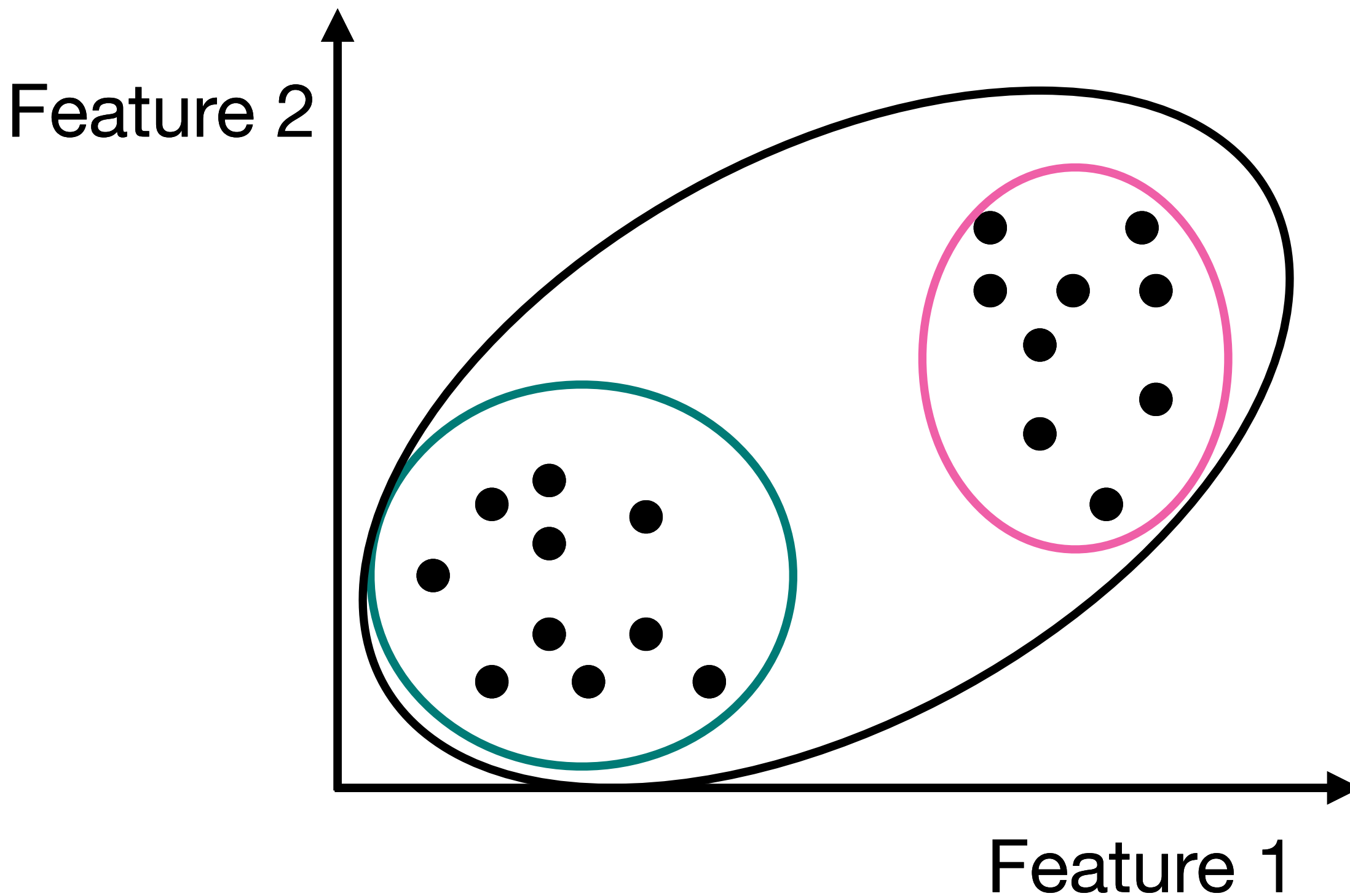
**There are 3 hyperparameters:**

3. The distance metric to calculate the distance between clusters e.g. Euclidean

$$d(\vec{x}_1, \vec{x}_2) = ||x_2 - x_1|| = \sqrt{\sum_{i=1}^n (x_1^i - x_2^i)^2}$$

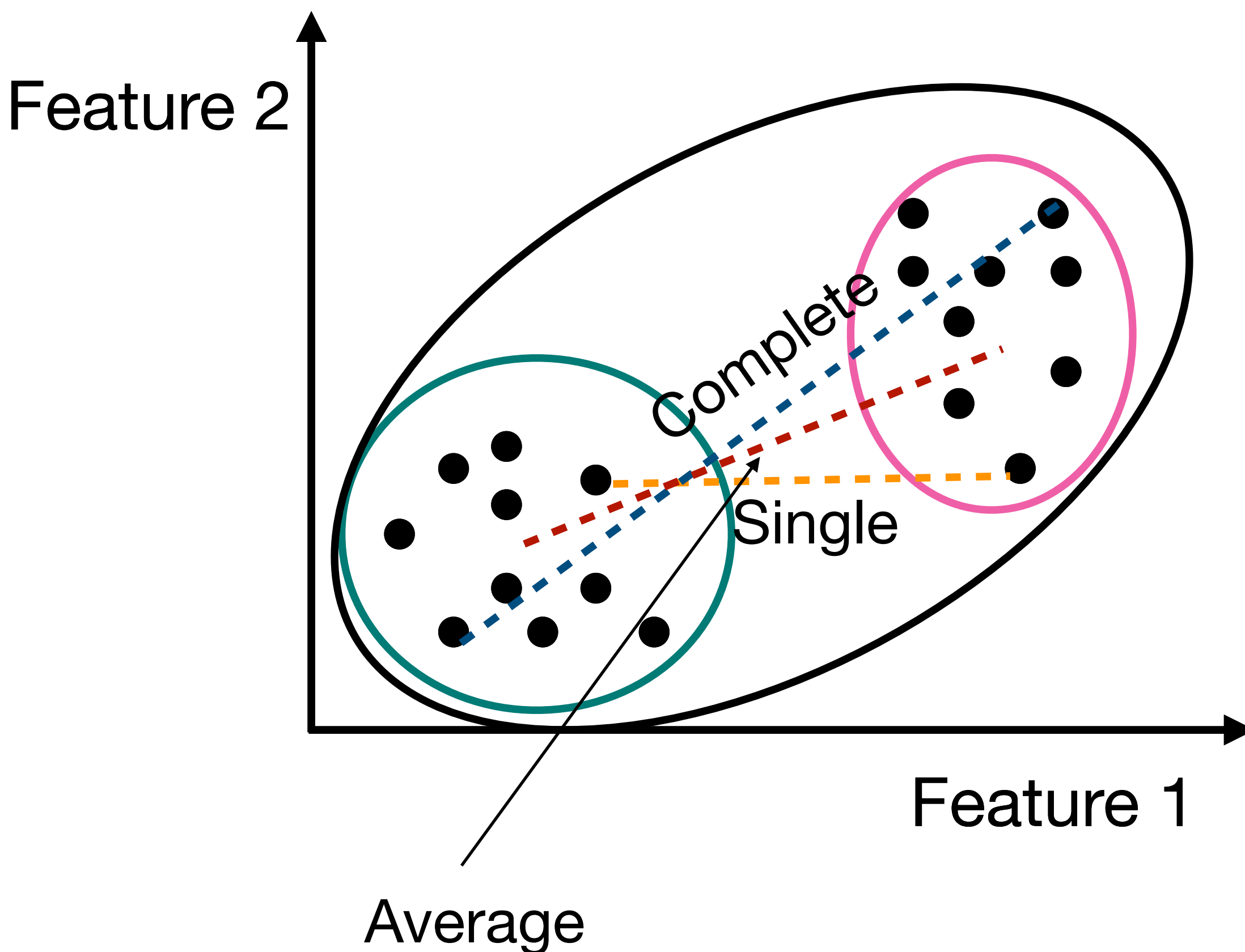
## Linkage method:

- Finding distances between clusters is something also to be tuned (find which works best for you)
- When clusters have multiple points, defining the “distance” between two clusters is non-trivial
- How we do this is known as our algorithm’s **linkage method**



## Examples of linkage methods:

- Single — distance between two closest points
- Average — distance between average cluster position
- Complete — distance between two furthest points
- Ward — keeps growth in sum of squares as small as possible (requires Euclidean metric)

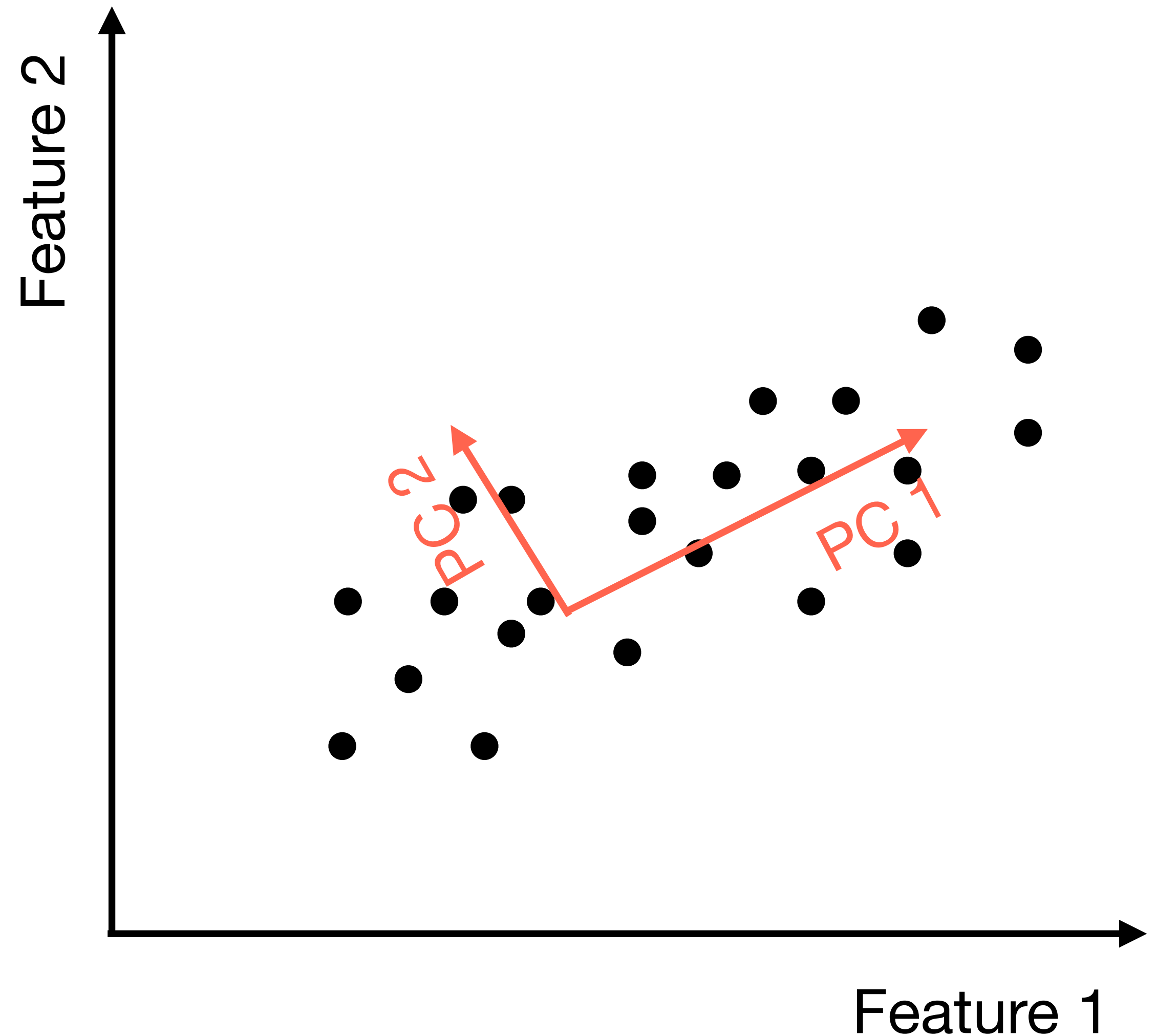


- Decomposition of data into important features and embedding a high-dimensional dataset into a lower-dimensional space.
- Important as certain features of your data may be redundant
- Improves performance of supervised learning
- Data compression/uncover complex trends



# Principal Component Analysis (PCA)

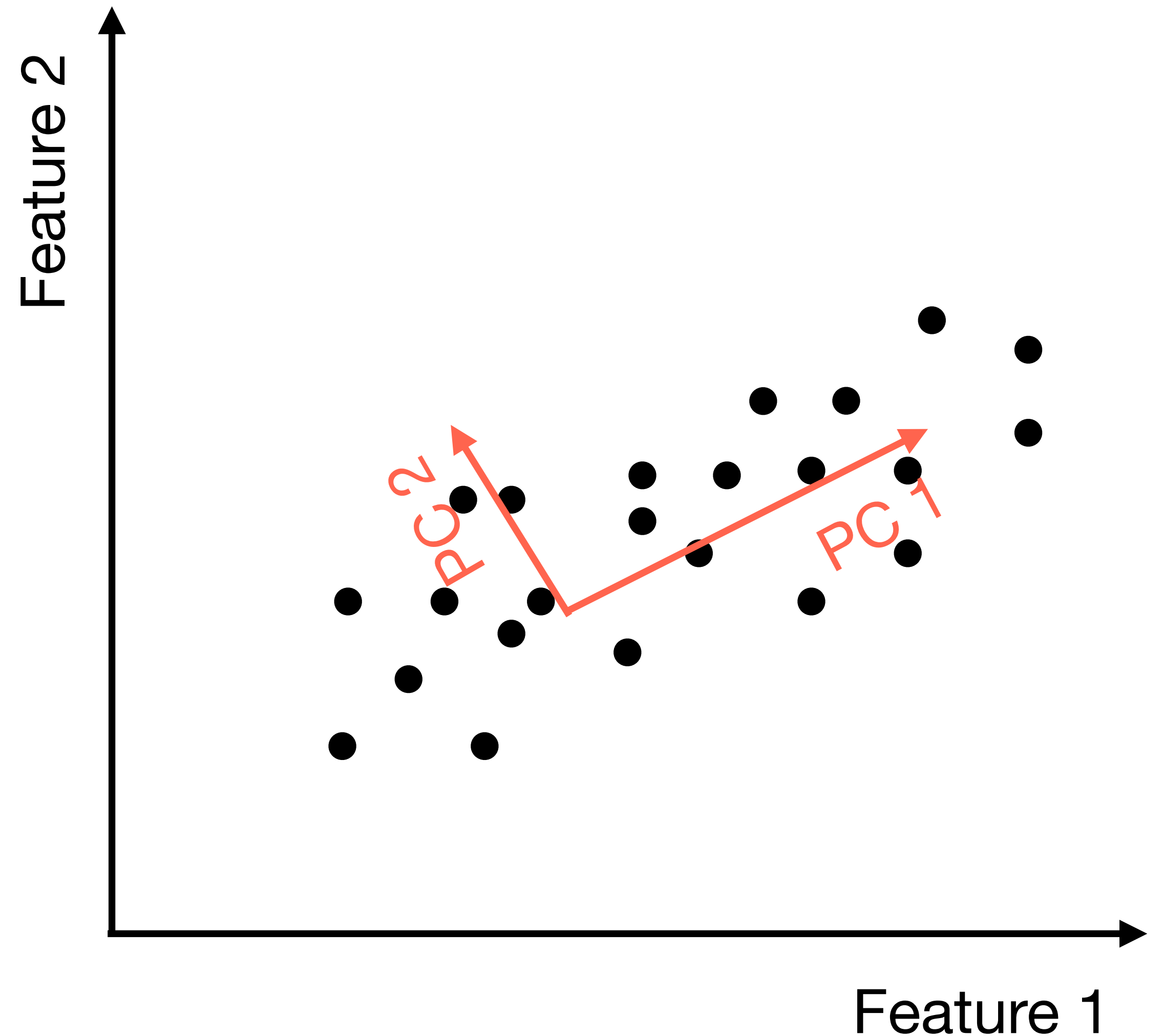
- Orthogonal linear transformation to basis of so-called “principal components”
- These are the directions in the data with the most variance





# Principal Component Analysis (PCA)

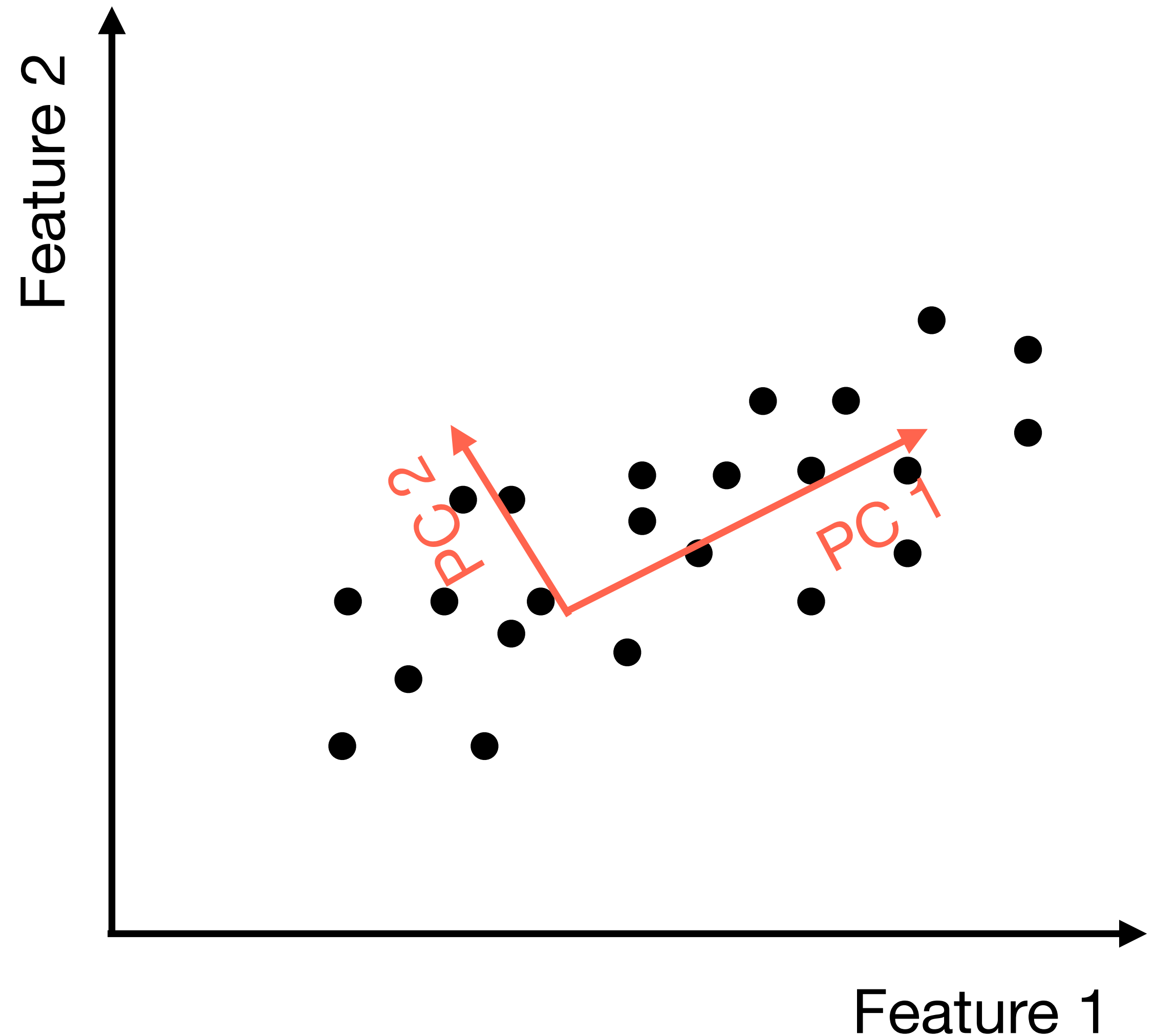
- The first principal component has the largest variance in the data
- Then each proceeding component has the largest variance of the remaining directions



# Principal Component Analysis (PCA)

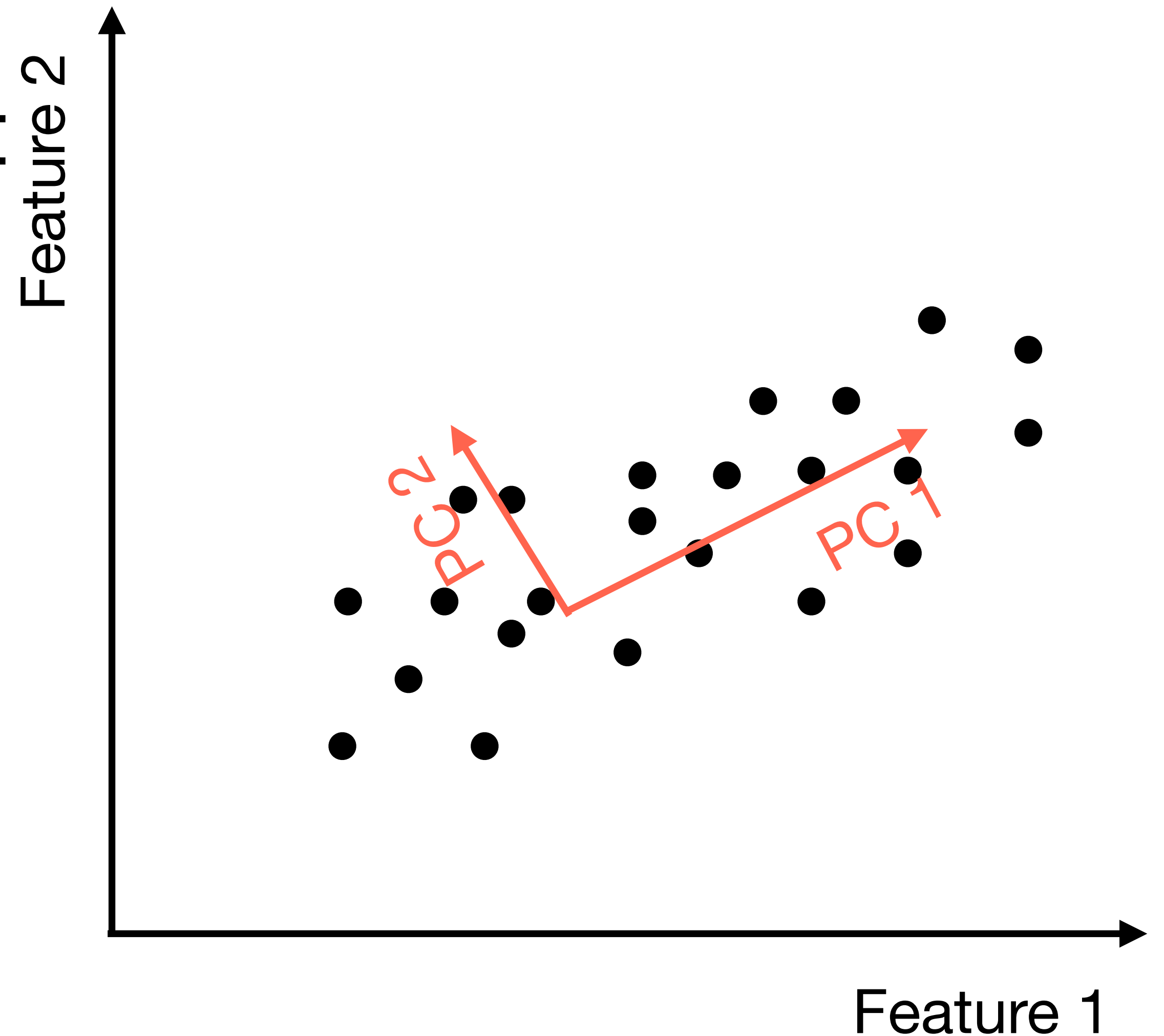
- PCA allows you to represent the data as a linear combination of the principal components

$$x = \sum_{i=1}^N A_i \times PC_i$$



# Principal Component Analysis (PCA)

- Data can then be compressed by expressing it in terms of the most important principal components
- This also gives a low-dimensional representation



# There are many other methods

- I have only explained two of a variety of techniques that can be used
- There are many other clustering and dimensionality reduction techniques to be used that may be more useful to your research
- Classical unsupervised learning can also be used for outlier detection and classification which we have not covered.
- <https://scikit-learn.org/> has good examples for many methods and an easy-to-use API
- sklearn also has a great cheatsheet: <http://blog.kaggle.com/2015/04/15/scikit-learn-video-2-setting-up-python-for-machine-learning/> on deciding what algorithm to use